Programmierung

13. Übungsblatt

Zeitraum: 20. – 24. Juli 2015

Übung 1 (AGS 16.25)

(a) Transformieren Sie die folgende Funktion `h` eines H_0-Programmes in ein AM_0-Programm mit baumstrukturierten Adressen. Geben Sie dabei keine Zwischenschritte an.

```
h :: Int -> Int -> Int -> Int
h x1 x2 x3 = if x3 > x1 then x2 - 1 else h x2 (x1 - x3) x2
```

(b) Folgendes H_0-Programm sei gegeben:

```
module Main where

h :: Int -> Int -> Int
h x1 x2 = if x2 == x1 then 30
                      else x2

g :: Int -> Int -> Int
g x1 x2 = if 10 <= x2 then g (x1-x2) (x2-1)
                      else h (x1+x2) 10

main = do x1 <- readLn
          print (g (3+x1) 5)
```

Vervollständigen Sie die Angaben `/*A*/` bis `/*F*/` in der folgenden Übersetzung des H_0-Programms in ein äquivalentes C_0-Programm:

```
#include <stdio.h>

int main() {
  int x1, x2, function = 2, flag, result;
  /*A*/
  while (flag == 1) {
    if (function == 1)
      if (/*B*/) {
        /*C*/
      }
      else {
        /*D*/
      }
    else if (/*E*/) {
      /*F*/
    }
  }
  printf("%d", result);
  return 0;
}
```

Übung 2 (AGS 16.20)

(a) Schreiben Sie ein C_0-Programm auf, das durch Anwendung der aus der Vorlesung bekannten Transformationsfunktion in das folgende H_0-Programm überführt werden kann (auf die Typdefinition wurde hier verzichtet):

```
module Main where

f1    x1 x2 = f2 x1 x1
f2    x1 x2 = if x2 > 0 then f21 x1 x2 else f3 x1 x2
f21   x1 x2 = f211 x1 x2
f211  x1 x2 = f212 (x1*x2) x2
f212  x1 x2 = f2 x1 (x2-1)
f3    x1 x2 = x1

main = do x1 <- readLn
          print (f1 x1 0)
```

(b) Folgendes H_0-Programm sei gegeben:

```
module Main where

f :: Int -> Int -> Int
f x1 x2 = if x2 > 0 then f (x1*x2) (x2-1) else g x1 (x1*x1)

g :: Int -> Int -> Int
g x1 x2 = if x1 < 100 then x2 else x1

main = do x1 <- readLn
          print (f x1 (2*x1))
```

Geben Sie in der folgenden Übersetzung des H_0-Programms in ein äquivalentes C_0-Programm für `/*A*/` bis `/*F*/` die entsprechenden *C*-Ausdrücke an:

```
#include <stdio.h>

int main() {
  int x1, x2, function, flag, result;
  /*A*/
  while (flag == 1) {
    if (function == 1)
      if (/*B*/) {
        /*C*/
      }
      else {
        /*D*/
      }
    else if (/*E*/) {
      /*F*/
    }
  }
  printf("%d", result);
  return 0;
}
```

Zusatzaufgabe 1 (AGS 14.17)

(a) Gegeben sei folgendes Fragment eines C_1-Programms mit den Funktionen `f` und `g`:

```
while(*p > i) {
  f(p);
  i = i + 1;
}
p = &i;
```

Übersetzen Sie die Sequenz dieser Statements in entsprechenden AM_1-Code mit baumstrukturierten Adressen (mittels *stseqtrans*). Sie müssen keine Zwischenschritte angeben. Nehmen Sie an, die `while`-Anweisung sei das zweite Statement in `g`, und es sei

$$tab_{g+lDecl} = \{\mathtt{f}/(\text{proc},1), \mathtt{g}/(\text{proc},2), \mathtt{i}/(\text{var},\text{lokal},1), \mathtt{p}/(\text{var-ref},-2)\}\,.$$

(b) Gegeben sei folgender AM_1-Code:

```
1: INIT 1;          8: LIT 2;          15: STORE(lokal,-2); 22: LOAD(global,1);
2: CALL 18;         9: LOADI(-3);      16: JMP 4;           23: PUSH;
3: INIT 0;         10: MUL;            17: RET 2;           24: CALL 3;
4: LOAD(lokal,-2); 11: STOREI(-3);     18: INIT 0;          25: WRITE(global,1);
5: LIT 0;          12: LOAD(lokal,-2); 19: READ(global,1);  26: JMP 0;
6: GT;             13: LIT 1;          20: LOADA(global,1);
7: JMC 17;         14: SUB;            21: PUSH;
```

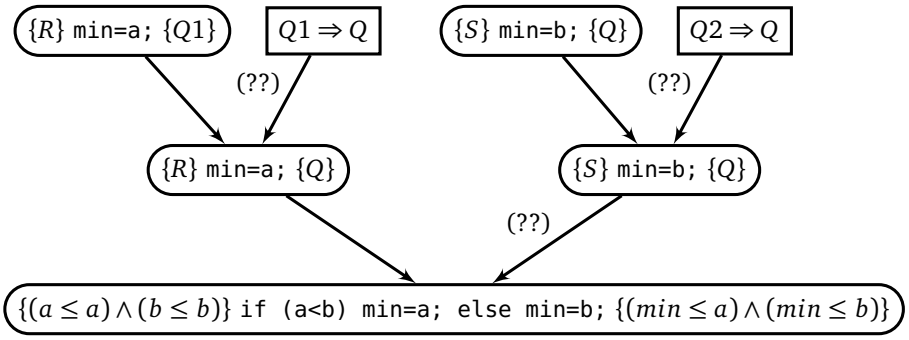
Führen Sie die AM_1 auf der Konfiguration $\sigma = (22, \varepsilon, 1:3:0:1, 3, \varepsilon, \varepsilon)$ weiter. Sie müssen nur 13 Schritte ausführen!

Zusatzaufgabe 2 (AGS 15.12 ⋆)

Mit Hilfe des Hoare-Kalküls wurde für die Verifikationsformel

$$\{(a \le a) \wedge (b \le b)\}\texttt{if (a<b) min=a; else min=b;}\{(min \le a) \wedge (min \le b)\}$$

der folgende korrekte Beweisbaum aufgestellt:



Geben Sie für R, S, Q, $Q1$, $Q2$ die konkreten Zusicherungen an. Des Weiteren nennen Sie alle angewendeten Verifikationsregeln und geben Sie explizit die an den Blättern des Beweisbaumes auftretenden Instanzen des Zuweisungsaxioms an.

Zusatzaufgabe 3 (AGS 12.4.17)

(a) Reduzieren Sie den untenstehenden λ-Term, bis seine Normalform erreicht ist. Schreiben Sie – bevor Sie einen Ableitungsschritt ausführen – für die relevanten (Teil-)Ausdrücke die Mengen der frei bzw. gebunden vorkommenden Variablen auf.

$$(\lambda zx.x\ z(\lambda y.\ yx\ ))(\lambda y.zx)\,z$$

(b) Eine Funktion $g : \mathbb{N} \times \mathbb{N}^+ \to \mathbb{N}^+$ sei wie folgt definiert:

$$g(x,y) = \begin{cases} y\,, & \text{wenn } x = 0\,, \\ y * g(x-1, y+1)\,, & \text{wenn } x > 0 \text{ und } x \text{ ist gerade}\,, \\ 2 + g(x-1, y)\,, & \text{wenn } x > 0 \text{ und } x \text{ ist ungerade}\,. \end{cases}$$

Geben Sie zur Funktion g den zugehörigen λ-Term $\langle G \rangle$ an, so dass $\langle g \rangle = \langle Y \rangle \langle G \rangle$ gilt.

(c) Gegeben sei der λ-Term:

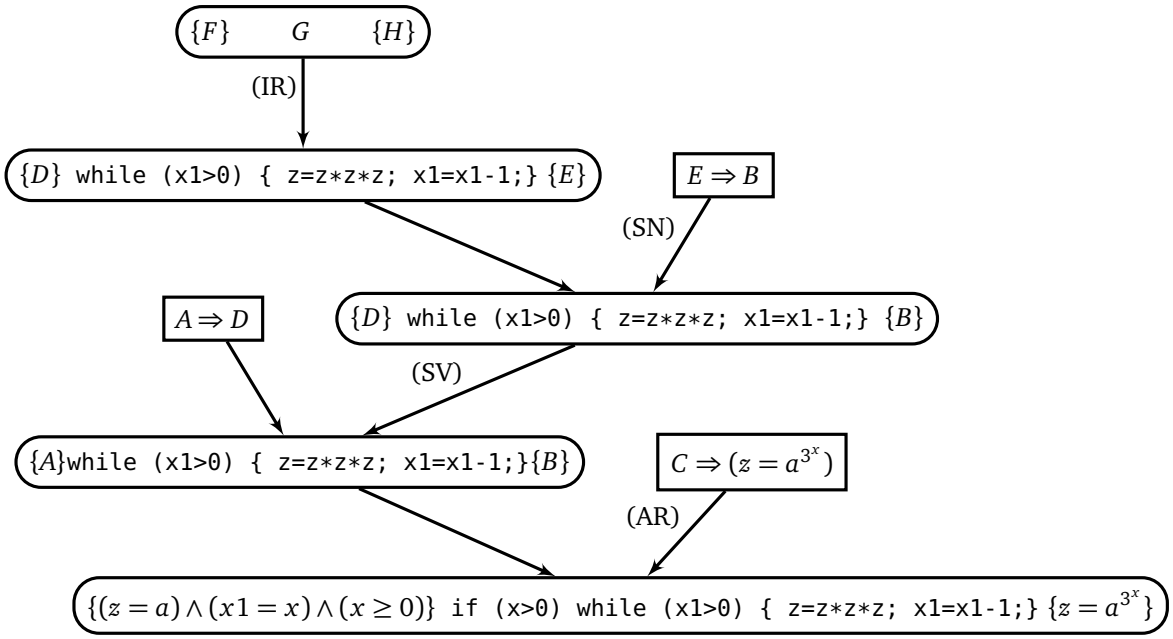
$$\langle F \rangle = \big(\lambda f z y x.\langle ite \rangle \big(\langle iszero \rangle\ y\big)\ (\langle mult \rangle x\ z) \big(f\ (\langle succ \rangle z)\ (\langle pred \rangle y)\ (\langle succ \rangle (\langle succ \rangle x))\big)\big)$$

Berechnen Sie die Normalform des Terms $\langle Y \rangle \langle F \rangle \langle 5 \rangle \langle 1 \rangle \langle 3 \rangle$. Führen Sie im Rechenprozess zweckmäßige Abkürzungen der λ-Terme ein.

Zusatzaufgabe 4 (AGS 15.9)
Für die Verifikationsformel

$$\{(z = a) \wedge (x1 = x) \wedge (x \geq 0)\}\ \texttt{if (x>0) while (x1>0) \{ z=z*z*z; x1=x1-1;\}}\{z = a^{3^x}\}$$

wurden die ersten vier (korrekten) Regelanwendungen des Beweisbaums aufgeschrieben. Im Term a^{3^x} soll dabei die „obere“ Exponentiation 3^x höhere Priorität haben. Die Ausdrücke A bis H sind noch unbekannt.



(a) Geben Sie eine geeignete Schleifeninvariante an.

(b) Geben Sie die Ausdrücke A, B, C, D, E, F, G und H an. Kürzen Sie gegebenenfalls die Schleifeninvariante mit SI ab.